ADAGEPA: ADAPTIVE FEEDBACK ALLOCATION FOR REFLECTIVE PROMPT OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Prompt optimization improves the performance of language-model systems on downstream tasks by refining their prompts. Classical methods evaluate prompts on task examples and use the resulting feedback to guide prompt updates through reflection. However, when feedback selection does not account for the prompt's weaknesses, these updates may improve performance on selected examples without yielding broader task improvements. To address this issue, we propose AdaGEPA, an adaptive feedback-allocation method that uses the prompt's performance and task structure to select examples for the next prompt update. Our method replaces at most one example in each feedback minibatch to target an identified weakness while preserving the remaining feedback context. Across our main experiments on six downstream benchmarks, AdaGEPA achieves higher mean validation scores than non-adaptive feedback selection under matched rollout budgets. AdaGEPA also finds high-performing prompts earlier across several tasks. In the initial Schema-Guided Dialogue (SGD) study, its half-budget prompts outperform the non-adaptive baseline's full-budget prompts in joint goal accuracy on new dialogues from services seen and unseen during search. Overall, our findings highlight the potential of adaptive feedback allocation to improve both the effectiveness and rollout efficiency of reflective prompt optimization.

1 INTRODUCTION

Prompt optimization adapts language-model systems to downstream tasks by refining their instructions without updating model weights (Yang et al., 2024; Pryzant et al., 2023). These tasks include task-oriented dialogue (Rastogi et al., 2020; Budzianowski et al., 2018), mathematical reasoning (Dekoninck et al., 2026), instruction following (Pyatkin et al., 2025), and multi-hop verification (Jiang et al., 2020). Beyond scalar outcomes, task executions can provide richer evidence, including reasoning traces, tool outputs, and evaluator feedback that can reveal why an output failed. Reflective prompt optimizers use this evidence to guide subsequent prompt revisions. Agrawal et al. (2025) introduced GEPA, a reflective prompt optimizer that maintains a pool of candidate prompts and uses natural-language reflection to revise a selected parent based on feedback from a minibatch of task examples. Because the examples in that minibatch determine which evidence shapes the next prompt update, their selection can affect both the revised candidate and the subsequent search trajectory. This raises a natural question: how should feedback examples be selected to address the current prompt's weaknesses and improve performance beyond the selected minibatch?

The GEPA default shuffles feedback examples without targeting the selected parent's weaknesses. A natural alternative is to resample the entire minibatch to favor examples with larger loss gaps, repeated failures, or underrepresented task groups. However, our initial experiments reveal a gap between early progress and the final search outcome: although this strategy can improve early candidate quality and produce more candidates that improve on their parents, it does not consistently improve the best validation score found by the end of search. One challenge is that reflection turns feedback from a small minibatch into a prompt revision that affects behavior across the task. A revision that fixes the selected errors may leave other errors unresolved or weaken behavior that was already correct. This tension echoes two established observations: example choice and ordering matter in curriculum learning, while a shared update can help one behavior but harm another in multi-task learning (Bengio et al., 2009; Yu et al., 2020). These observations suggest that difficulty and coverage alone do not reliably identify feedback that leads to broader task improvements. We

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

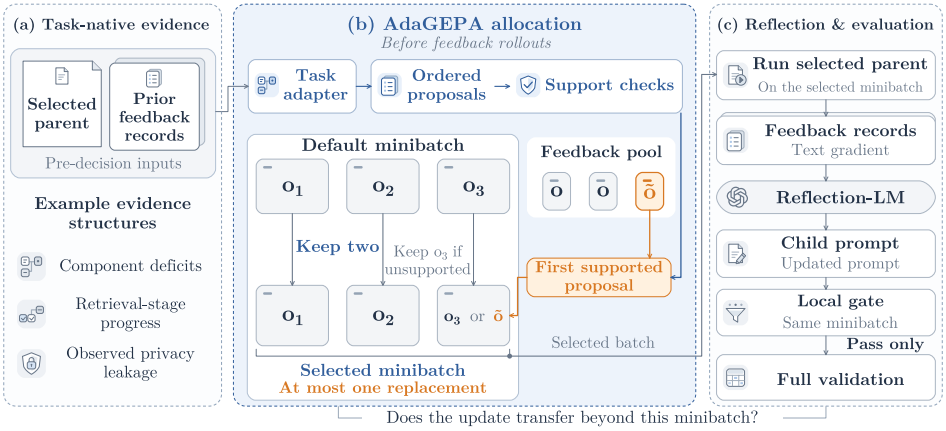

Figure 1: AdaGEPA adapts the feedback minibatch before reflection. (a) Cached evaluation records provide task-specific signals about the selected parent and search state. (b) A task adapter prioritizes eligible proposals and replaces at most one example in the default minibatch. If none passes the support checks, the default minibatch is retained. (c) The selected minibatch supplies feedback for the next prompt update. The resulting candidate is compared with its parent on that minibatch and proceeds to full validation if its local score improves.

therefore study how task structure and the parent prompt's performance can guide feedback selection toward revisions that benefit examples beyond the minibatch.

In this paper, we study pre-reflection feedback allocation and propose AdaGEPA, a lightweight method that uses cached evaluation records and task structure to select examples for the next prompt revision. Figure 1 shows how AdaGEPA makes this choice after parent selection and before feedback generation. Depending on the task, this structure may describe service categories, evaluation components, or intermediate execution stages. Rather than resampling the entire minibatch, AdaGEPA replaces at most one example in the default minibatch. Retaining the other examples preserves most of the default feedback context while adding targeted evidence about the prompt's weaknesses. If no eligible replacement is available, AdaGEPA uses the default minibatch.

For example, in dialogue state tracking, cached evaluations can reveal whether the current parent struggles with intent prediction or slot-value tracking. A task-specific adapter can then propose an example whose task profile is aligned with those weaknesses. The replacement is applied only if it satisfies the task-specific eligibility and scheduling rules. The selected examples then provide feedback for the prompt update. The resulting candidate is evaluated on the same minibatch and, if it improves on the parent, proceeds to full validation to determine whether it improves the best score found so far.

Our contributions are summarized as follows:

- We formalize pre-reflection feedback allocation as a distinct decision in reflective prompt search: after parent selection, the optimizer chooses which examples inform reflection. We define its immediate utility as the expected one-step gain in the best-so-far validation score.
- We propose AdaGEPA, a lightweight method that uses task structure and cached evaluations to target feedback to a prompt's weaknesses. It replaces at most one minibatch example and keeps the default minibatch unchanged if no replacement is eligible. Feedback selection adds no language-model calls, and subsequent reflection and evaluation remain unchanged.
- Across six downstream benchmarks, AdaGEPA yields higher mean final validation scores than default feedback selection at matched budgets and finds high-scoring prompts earlier. In the initial Schema-Guided Dialogue (SGD) study, its half-budget prompts outperform the full-budget baseline on new dialogues involving both previously seen and unseen services.
- Through task-specific controls and trajectory analyses, we trace feedback choices through candidate evaluation, best-score improvements, and the parent-child path to the final winner (winner

lineage). This shows how early candidate improvements relate to final outcomes and whether they persist as search continues.

2 Related Work

Textual feedback and prompt optimization. Prompt optimizers differ in how they use evaluation evidence. OPRO uses earlier candidates and objective values to guide new proposals. EvoPrompt and Promptbreeder combine LM-generated mutations with task-level fitness, while MIPRO uses stochastic minibatches and a surrogate objective to optimize instructions and demonstrations (Yang et al., 2024; Guo et al., 2024; Fernando et al., 2023; Opsahl-Ong et al., 2024). Other methods use feedback to shape prompt revisions directly. ProTeGi derives natural-language critiques from minibatch examples, TextGrad propagates textual feedback through compound computation graphs, and GEPA reflects on task-native feedback and execution trajectories within a Pareto-based search (Pryzant et al., 2023; Yuksekgonul et al., 2024; Agrawal et al., 2025). Our work focuses on the decision before reflection: which examples should inform the next prompt revision. We then examine whether that revision improves performance beyond the selected examples.

Data selection in prompt search. Data selection enters prompt search at several decision points. During optimization, APEX uses prompt-lineage history and solvability to choose mutation data, and rank sensitivity to choose candidate-evaluation data. Its 2×2 ablation examines these choices separately (Wang et al., 2026). TwinBandit jointly selects challenging instances and mutation strategies through two bandits (Park et al., 2025), whereas IPOMP updates the candidate-evaluation subset as optimization proceeds (Dong et al., 2025). Other methods select data before the search begins. SESS constructs a representative submodular evaluation subset under OPRO (Nian et al., 2026), and p1 filters user prompts by separating prompt-quality variation from response stochasticity (Gao et al., 2026). For reflective updates, AdaGEPA uses the prompt's performance and task structure to select the examples that guide the next prompt revision.

Downstream outcomes and search control. Downstream outcomes can guide individual edits or broader search decisions. Learning to Self-Evolve rewards context edits by their improvement on a holdout set and trains an editing policy with reinforcement learning (Chen et al., 2026). ReASearch lets an agent choose prompt and program revisions, minibatches, evaluation budgets, and stopping decisions (Li et al., 2026). RLMOpt and OEO also place candidate revision, evaluation, and search continuation under the optimizer's control (Satheesha et al., 2026; Xue & Yang, 2026). DIVE instead evolves reusable skill artifacts through diversity-driven search (Xiong et al., 2026). We focus on an earlier, narrower decision: which examples inform reflection. We examine whether the resulting candidate raises the best full-validation score and whether later search builds on it.

3 Preliminaries

3.1 Prompt Optimization under a Rollout Budget

We study prompt optimization for a compound AI system under a task-level rollout budget. We use Φ to denote a candidate system instantiated with a particular prompt configuration. Model weights and all non-prompt components remain fixed across candidates (Agrawal et al., 2025). A task instance is a pair (x, m), where x is the system input and m contains the information used to evaluate its output. The task metric $\mu(\Phi(x), m) \in [0, 1]$ returns a scalar score, and the feedback function μ_f supplements it with task-native textual feedback for reflection.

We measure this budget in rollouts. A rollout consists of running a candidate Φ on one task instance and evaluating its output with μ. The optimization budget B limits how many of these task-level evaluations may be performed. Calls to models and tools, including the language model used for reflection (Reflection-LM), are counted separately.

3.2 Reflective Search with Scheduled Feedback

During development search, the optimizer uses a feedback set $\mathcal{D}_{\text{feedback}}$ and a validation set $\mathcal{D}_{\text{pareto}}$. At the start of each epoch, the examples in $\mathcal{D}_{\text{feedback}}$ are randomly shuffled and then consumed in

consecutive minibatches. We call each scheduled use of a task example a *feedback occurrence*, so repeated uses of the same example count as distinct occurrences. Running a candidate on the selected occurrences produces execution records and evaluator feedback for reflection. $\mathcal{D}_{\text{pareto}}$ supports full candidate evaluation, parent selection, and winner selection.

At proposal step t, $\mathcal{P}_t$ contains the candidates that have completed full evaluation on $\mathcal{D}_{\text{pareto}}$. Parent selection uses their instance-level scores, so a candidate that leads on some instances can remain eligible even without the highest average score (Agrawal et al., 2025).

After an existing candidate Φ_k is selected as the parent, the default feedback schedule supplies the next b occurrences to form $\mathcal{M}_t^0$. Running the parent on these occurrences produces execution records and evaluator feedback. The Reflection-LM uses this information to revise one prompt component of the parent candidate, producing a new candidate Φ', called the child. The resulting child is then evaluated on the same minibatch as its parent. If its average score is higher than the parent's, it is fully evaluated on $\mathcal{D}_{\text{pareto}}$ and added to the candidate pool. Parent choice and prompt revision therefore adapt during search, but the feedback minibatch does not depend on the selected parent's current state. Appendix B.2 provides additional details of this search transition.

3.3 Pre-Reflection Feedback Allocation

Feedback allocation occurs after parent selection but before any rollout is executed on the next feedback minibatch. The chosen occurrences therefore determine which execution records and evaluator feedback serve as input to the Reflection-LM. We call this decision *pre-reflection feedback allocation*. At proposal step t, let ξ_t denote the pre-reflection search state, including the selected parent, the default minibatch $\mathcal{M}_t^0$, cached evaluation records, and the state of the feedback schedule. Let $\mathbb{A}(\xi_t)$ denote the feedback-allocation actions available in this state. An action $a_t \in \mathbb{A}(\xi_t)$ selects b occurrences and produces $\mathcal{M}_t(a_t)$. The default action a_0 leaves $\mathcal{M}_t^0$ unchanged, so $\mathcal{M}_t(a_0) = \mathcal{M}_t^0$. Across actions, the selected parent, minibatch size, rollout cap, and all subsequent search rules remain fixed. The action is chosen using only information available in ξ_t, before the resulting child or any later search outcome is observed.

4 Methods

4.1 Motivation and Design Rationale

Feedback selection should consider not only whether the resulting prompt improves the examples used for reflection, but also how it performs on the full validation set. The local gate admits a child when it improves on its parent over the selected minibatch (Agrawal et al., 2025). However, a child that passes this gate need not outperform its parent under full validation (Appendix A). Even an improvement over the parent may leave the best-so-far score unchanged. Therefore, feedback allocation must distinguish local repair, parent-level improvement, and frontier progress.

One simple alternative is to resample the entire minibatch using loss gaps, repeated failures, and coarse task coverage. Our design studies show that this history-and-coverage rule can improve early candidate quality, but its gains do not consistently persist as search continues (Appendices C.2.1 and C.3.3). Therefore, difficulty and coverage can identify promising feedback, but cannot tell us whether the resulting revision will improve full-validation performance.

To address this limitation, our proposed method, AdaGEPA, aims to combine task-native structure with cached evaluations of the selected parent. Depending on the task, this structure may describe behaviors, components, or stages, while the cached records reveal where the parent performs poorly. A task adapter uses this evidence to propose one replacement for the default minibatch. By keeping the remaining examples unchanged, this one-slot design preserves most of the default feedback context while directing one position toward a task-relevant weakness.

4.2 Immediate Frontier Utility

We define the immediate utility of a feedback choice by how much the resulting prompt update advances the best validation score found so far. Let $\bar{S}(\Phi)$ denote the average score of candidate Φ on the full validation set $\mathcal{D}_{\text{pareto}}$, and let H_t be the highest score in the candidate pool at the start of

proposal step t:

$$\bar{S}(\Phi) = \frac{1}{|\mathcal{D}_{\text{pareto}}|} \sum_{(x_i, m_i) \in \mathcal{D}_{\text{pareto}}} \mu(\Phi(x_i), m_i), \quad H_t = \max_{\Phi \in \mathcal{P}_t} \bar{S}(\Phi).$$

Given a feedback choice $a \in \mathbb{A}(\xi_t)$, let $H_{t+1}^{(a)}$ be the best-so-far score after applying a in state ξ_t and completing the resulting proposal transition. We then define its one-step frontier gain, expected gain, and advantage over the default action a_0 as

$$g_t(a) = H_{t+1}^{(a)} - H_t, \quad U(a \mid \xi_t) = \mathbb{E}[g_t(a) \mid \xi_t], \quad \text{Adv}(a \mid \xi_t) = U(a \mid \xi_t) - U(a_0 \mid \xi_t).$$

Even from the same state and with the same feedback choice, stochastic reflection and candidate evaluation can produce different frontier gains. $U(a \mid \xi_t)$ is the expected gain over these possible outcomes. $\text{Adv}(a \mid \xi_t)$ measures the difference between this value and the expected gain of the default action a_0.

4.3 Bounded Feedback Allocation

AdaGEPA limits each action to at most one replacement, preserving most of the default feedback context. Let the default minibatch $\mathcal{M}_t^0$ have size b. The first $b-1$ occurrences remain fixed, and only the final occurrence may be replaced:

$$\mathcal{M}_t(a_0) = (o_{t,1}^0, \ldots, o_{t,b-1}^0, o_{t,b}^0), \quad \mathcal{M}_t(a) = (o_{t,1}^0, \ldots, o_{t,b-1}^0, \tilde{o}_{t,b}(a)).$$

We use $b = 3$ in all experiments, so the first two occurrences remain unchanged and only the third may be replaced. The replacement is chosen in three steps. **(1) Propose.** A task adapter uses the evidence available in ξ_t to nominate possible replacement occurrences and rank them according to a fixed priority rule. **(2) Check.** A proposal is supported only if the proposed occurrence belongs to $\mathcal{D}_{\text{feedback}}$, is distinct from every occurrence in the default minibatch, and satisfies the adapter-specific schedule constraints. Multiple proposals that name the same occurrence are treated as a single action. **(3) Select.** AdaGEPA selects the first supported proposal in priority order. If none is supported, it applies the default action a_0 and leaves $\mathcal{M}_t^0$ unchanged. Feedback selection does not consume the task-level rollout budget. Before a proposal begins, the runner verifies that enough budget remains to compare the parent and child on the minibatch and to fully evaluate the child if it passes the local gate.

Algorithm 1 summarizes this shared selection procedure. The task adapter A supplies an ordered list of proposals, while the support check and default fallback are shared across tasks.

Algorithm 1: AdaGEPA feedback selection

Input : selected parent Φ_k, default minibatch $\mathcal{M}_t^0$, cached records C, feedback set $\mathcal{D}_{\text{feedback}}$, schedule state q, frozen task adapter A.
Output : selected batch $\mathcal{M}$ and updated schedule state q'.

$L \leftarrow A.\text{propose}(\Phi_k, \mathcal{M}_t^0, C, \mathcal{D}_{\text{feedback}}, q)$
for each distinct occurrence o in L, in adapter priority order:
  if $\text{Supported}(o, \mathcal{M}_t^0, \mathcal{D}_{\text{feedback}}, q)$:
    $(\mathcal{M}, q') \leftarrow \text{ReplaceFinal}(\mathcal{M}_t^0, o, q)$
    return $(\mathcal{M}, q')$
return $(\mathcal{M}_t^0, q)$

Feedback selection uses only the cached records in ξ_t and adds no model calls. The parent then produces reflection records on the selected minibatch. The child is evaluated on this minibatch and proceeds to full validation only after passing the local gate.

4.4 Task-Specific Proposal Rules

Task adapters convert cached evaluation records, task metadata, and allocation history into prioritized replacement proposals. Depending on the task, the proposal rule can be a deterministic rule, a replay policy over eligible feedback-bearing occurrences, or a model fitted before evaluation. In each case, the evidence schema, proposal semantics, priority rule, and any fitted parameters are fixed before evaluation, while the actual proposals respond to the selected parent and available feedback.

For example, if cached evaluation records show that the selected parent performs poorly on a particular behavior, a task adapter can propose an eligible occurrence associated with that behavior. When several occurrences qualify, a fixed task-specific rule ranks them using the available search evidence. The relevant task structure can represent services, evaluation components, or stages. Appendix B.4 summarizes the task-specific evidence schemas and proposal rules used in our experiments.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

Evaluation design. Development-search curves track when each method discovers high-quality candidates and how the performance gap evolves with the rollout budget. We compare AdaGEPA with the GEPA default at matched checkpoints, namely candidates saved at fixed rollout budgets. In the original SGD study, we also compare AdaGEPA at $B = 560$ with the full-budget GEPA default. For SGD, MultiWOZ, and HoVer, we further evaluate frozen candidates on new examples (search-external panels), without further prompt optimization.

Tasks and common configuration. Our experiments separate design studies from evaluations of the final method. Full-minibatch history-and-coverage studies on SGD, AIME, and LiveBench-Math (White et al., 2025) test whether these signals are sufficient to guide feedback selection. We then evaluate AdaGEPA's bounded one-slot intervention across six tasks: dialogue state tracking (SGD and MultiWOZ), multi-hop verification (HoVer), instruction following (IFBench), privacy-sensitive generation (PUPA), and algorithmic code generation (TACO). The main paired experiments use feedback minibatches of three occurrences, `gpt-4.1-mini` to execute task examples (Task-LM), and `gpt-5.1` as the Reflection-LM. Within each comparison, the two arms share the initial candidate, rollout cap, and search procedure. They differ only in feedback selection. Matched controls isolate the effects of one-slot replacement, task-specific targeting, proposal-pool construction, and ordering. Task configurations, evaluation protocols, and additional results appear in Appendices B and C.

Metrics and statistics. Each task is evaluated using its task-specific metric. In the main AdaGEPA results, endpoint scores are reported as percentages and paired differences as percentage points. For dialogue state tracking, Balanced DST combines joint-state, intent, requested-slot, and slot-value scores. Joint goal accuracy (JGA) requires the complete state to be correct. For development searches, Gain-AUC measures the area between the best-so-far curve and its initial score over the normalized rollout budget. We pair runs by optimization seed and report mean scores and mean paired differences in the main tables. Appendix B.7 provides the metric definitions, median paired differences, win/tie/loss counts, 95% uncertainty intervals, repeated-evaluation aggregation, and run-inclusion rules.

5.2 MAIN RESULTS

The main experiments address two questions: whether adaptive feedback selection helps search discover higher-scoring prompts under the same rollout budget, and whether the resulting prompts retain their gains on examples not used during search. Figure 2 traces how best-so-far candidate quality changes over search, while Table 1 summarizes the earlier-checkpoint and endpoint comparisons. In development search, AdaGEPA reached a higher mean endpoint than the GEPA default on all six tasks. The clearest early gains appeared on SGD, MultiWOZ, and HoVer, while the smaller improvements on IFBench, PUPA, and TACO emerged later. Two additional SGD studies reproduced the early advantage, although the endpoint gaps narrowed as the rollout budget increased.

To test the second question, we evaluated frozen candidates from SGD, MultiWOZ, and HoVer on examples not used during development search. The clearest transfer result came from the original SGD study. Using half the optimization rollout budget, AdaGEPA found prompts that exceeded the GEPA default's full-budget prompts by 7.69 JGA points on new dialogues from previously seen services (in-domain) and 5.19 points on unseen services. The two additional SGD studies also favored AdaGEPA at $B = 560$ on both panels, although the endpoint gaps narrowed as the GEPA default continued to improve. Table 2 reports both Balanced DST and the stricter JGA for SGD and MultiWOZ. On MultiWOZ, AdaGEPA led by 1.56 Balanced DST points at $B = 560$ and 3.12 points at $B = 1120$. HoVer also showed positive mean search-external differences of 1.50

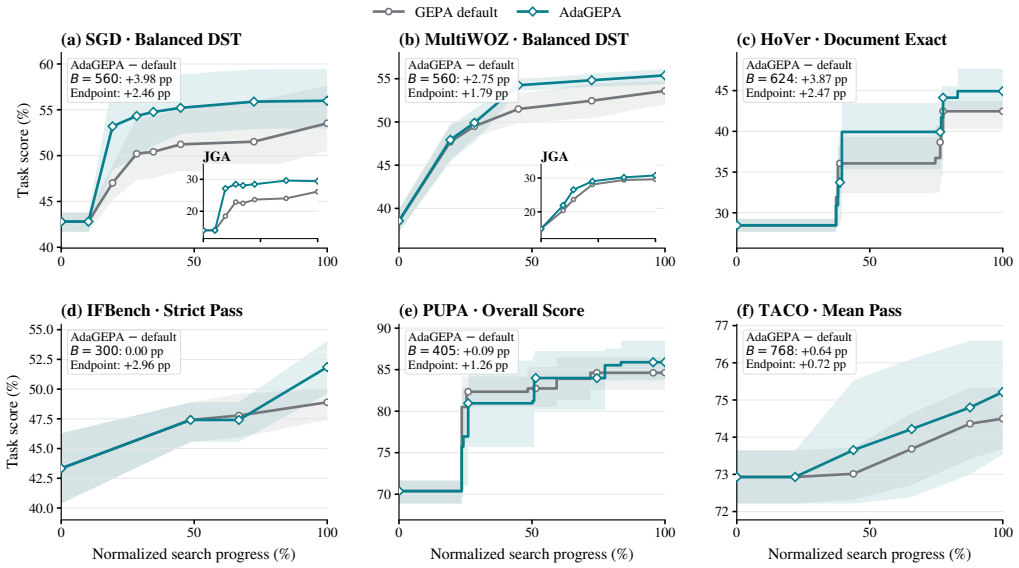

Figure 2: Development-search trajectories across six tasks. Curves show mean best-so-far task scores over normalized search progress, from the shared initial candidate at 0% to the final checkpoint at 100% within each task; shading shows 95% seed-resampling intervals. Each panel uses the metric named in its title. The SGD and MultiWOZ insets show JGA, which requires the complete dialogue state to be correct. Callouts report mean paired differences (AdaGEPA minus GEPA default) at the labeled earlier checkpoint and endpoint while retaining the corresponding absolute B values. The SGD panel shows the initial study. HoVer and PUPA use step curves because candidates become available at seed-dependent budgets.

Table 1: Development-search candidate quality. Entries are mean task scores over optimization seeds, reported as percentages. Parentheses give paired mean differences (AdaGEPA minus GEPA default) in percentage points. Earlier checkpoints are task-specific and listed by their absolute B values. The final column reports the paired difference in budget-normalized Gain-AUC over each seed's development-search trajectory.

			Earlier checkpoint			Endpoint			
Task / study	**Metric**	n	B	**GEPA default**	**AdaGEPA (Δ)**	B	**GEPA default**	**AdaGEPA (Δ)**	**Δ Gain-AUC**
SGD original	Balanced DST	5	560	51.23	**55.21** (+3.98)	1120	53.53	**55.99** (+2.46)	+3.49
	JGA		560	23.65	**28.46** (+4.81)	1120	26.15	**29.42** (+3.27)	
SGD extension	Balanced DST	5	560	51.80	**53.59** (+1.79)	1120	**55.59**	55.18 (−0.40)	+0.73
	JGA		560	23.65	**26.54** (+2.88)	1120	27.50	**28.27** (+0.77)	
SGD 10-run	Balanced DST	10	560	52.17	**53.74** (+1.58)	1120	**55.35**	55.33 (−0.03)	+0.99
	JGA		560	24.42	**26.54** (+2.12)	1120	**28.17**	**28.17** (0.00)	
MultiWOZ	Balanced DST	5	560	51.50	**54.25** (+2.75)	1120	53.59	**55.39** (+1.79)	+1.48
	JGA		560	28.08	**29.04** (+0.96)	1120	29.62	**30.77** (+1.15)	
HoVer	Document Exact	5	624	36.07	**39.93** (+3.87)	1120	42.47	**44.93** (+2.47)	+1.90
IFBench	Strict pass	5	300	**47.41**	**47.41** (0.00)	560	48.89	**51.85** (+2.96)	+1.23
PUPA	Overall score	5	405	83.90	**83.99** (+0.09)	607	84.64	**85.90** (+1.26)	−0.11
TACO	Mean pass	5	768	73.01	**73.65** (+0.64)	1424	74.50	**75.21** (+0.72)	+0.45

Document Exact points at $B = 624$ and 1.90 points at $B = 1120$. Full SGD checkpoint curves appear in Appendix C.1.1.

5.3 Ablation Studies

We focus the main component ablations on SGD, which provides the most complete set of matched controls. Controls on the remaining tasks examine the same design choices under different feedback structures, with full results reported in the appendix.

Table 2: Search-external candidate quality. Entries are mean task scores over optimization seeds, reported as percentages. Parentheses give paired mean differences (AdaGEPA minus GEPA default) in percentage points. Earlier-checkpoint budgets are task-specific and shown with the task labels. Dialogue-state-tracking rows report Balanced DST followed by JGA.

				Earlier checkpoint		Endpoint ($B = 1120$)	
Task / study	**Panel**	**Metric**	n	**GEPA default**	**AdaGEPA (Δ)**	**GEPA default**	**AdaGEPA (Δ)**
SGD original		Balanced DST		50.04	**56.95** (+6.91)	51.47	**56.85** (+5.38)
$B = 560$	In-domain	JGA	5	22.31	**31.15** (+8.85)	23.46	**31.73** (+8.27)
		Balanced DST		56.29	**61.03** (+4.74)	57.33	**59.69** (+2.36)
	Unseen service	JGA		28.08	**34.04** (+5.96)	28.85	**32.31** (+3.46)
SGD extension		Balanced DST		51.24	**53.85** (+2.61)	55.27	**55.40** (+0.13)
$B = 560$	In-domain	JGA	5	23.85	**27.88** (+4.04)	28.08	**29.62** (+1.54)
		Balanced DST		56.21	**59.88** (+3.67)	59.90	**60.40** (+0.50)
	Unseen service	JGA		26.99	**32.69** (+5.71)	32.12	**32.88** (+0.77)
SGD 10-run		Balanced DST		51.05	**54.24** (+3.19)	54.20	**55.24** (+1.04)
$B = 560$	In-domain	JGA	10	23.08	**28.65** (+5.58)	27.40	**29.62** (+2.21)
		Balanced DST		56.56	**57.67** (+1.11)	**59.04**	58.79 (−0.24)
	Unseen service	JGA		27.88	**29.90** (+2.02)	**31.25**	30.87 (−0.38)
MultiWOZ		Balanced DST		50.92	**52.47** (+1.56)	51.21	**54.33** (+3.12)
$B = 560$	Search-external panel	JGA	5	**26.92**	**26.92** (0.00)	25.77	**28.65** (+2.88)
HoVer							
$B = 624$	Search-external panel	Document Exact	5	40.32	**41.82** (+1.50)	44.74	**46.64** (+1.90)

n counts paired optimization runs. Panel construction and metric definitions appear in Appendix B.7; rollout checkpoints and full paired results appear in Appendices B.6 and C.1.

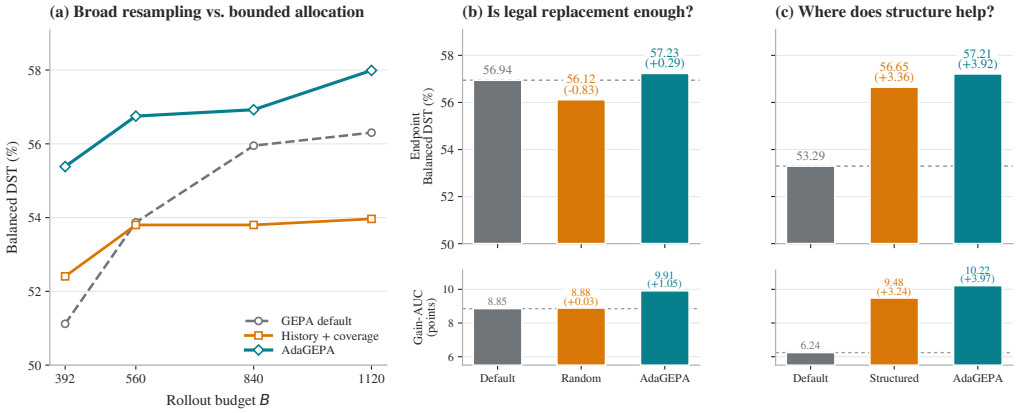

Figure 3: **Component ablations on SGD.** (a) Development best-so-far Balanced DST compares the GEPA default, full-minibatch history-and-coverage sampling, and AdaGEPA with ten formation runs. (b) Endpoint Balanced DST at $B = 1120$ (upper) and Gain-AUC (lower) compare the GEPA default, Random Legal, and AdaGEPA across ten optimization seeds. Random Legal selects a one-slot replacement uniformly from the legal feedback occurrences. (c) The same summaries compare AdaGEPA with Structured Unordered, which retains the supported replacement pool and fallback but removes fitted ordering. Parentheses report paired mean differences from the matched GEPA default. Full paired comparisons and additional controls appear in Appendix C.2.2.

We first examine how much of the feedback minibatch should be changed. On SGD, full-minibatch history-and-coverage sampling improved early candidate quality but stalled as the GEPA default continued to improve, motivating a more bounded intervention (Figure 3a). On AIME, replacing the full minibatch with a coarse selection rule left an endpoint deficit of 1.6 correct answers relative to the GEPA default. Applying the same rule to only one slot reduced this gap to 0.4. The one-slot constraint therefore bounds the intervention by introducing targeted feedback while preserving most of the default minibatch and its feedback context.

We next examine which components make a one-slot replacement useful. Random Legal keeps the one-slot constraint and samples uniformly from the legal alternatives. This control tests whether a legal replacement alone is sufficient without task-specific selection. Its mean Gain-AUC differed from the GEPA default by only $+0.03$ points, and its endpoint was 0.82 points lower. Structured Unordered keeps AdaGEPA's supported proposal pool and fallback but omits fitted ordering. Even without the ordering, it retained most of AdaGEPA's mean gain over the default. AdaGEPA exceeded Random Legal by 1.02 Gain-AUC points and 1.12 endpoint points. These comparisons suggest that one-slot replacement bounds the change to the feedback context but does not by itself improve search. Most of the observed improvement was recovered by structuring the proposal space. The additional gain from fitted ordering was smaller and varied across configurations. Full paired results and additional controls are reported in the appendix.

Finally, we examine which aspects of the parent's performance should guide feedback selection. On SGD, AdaGEPA refines service-level selection using intent and slot deficits. Its mean unseen-service JGA exceeded the service-level-only control by 4.81 points at $B = 560$ and 0.38 points at $B = 1120$ (Table 21). On MultiWOZ, selecting examples from services where the parent performed poorly yielded a mean endpoint Balanced DST 2.15 points above uniform selection of unused examples. However, adding fitted ordering within those services reduced mean endpoint quality (Appendix C.2.4). Across these controls, task-specific targeting provided the main improvement, while the value of finer ordering depended on the task and configuration.

6 Discussion

The development curves suggest that feedback selection may matter most when useful feedback is hard to reach within the search budget. AdaGEPA can find useful candidates early, while its lead sometimes narrows as search continues. One possible explanation is that task-native selection brings relevant examples into reflection before the shuffled schedule reaches them. This leads to a testable hypothesis: holding the task and adapter fixed, AdaGEPA should help most when the default schedule can cover only a small part of the feedback pool within the search budget.

Selecting a relevant example does not ensure that the resulting prompt update will remain useful through later search. On MultiWOZ, targeting services where the current parent performed poorly yielded higher development scores than choosing unused examples without service information. PUPA illustrates the next hurdle. Even when feedback from observed privacy failures was reused, most resulting updates did not appear in winner lineages. A child can enter the candidate pool without setting a new best score and may later be selected as the parent for another revision. Therefore, a fuller assessment of feedback allocation should consider both the immediate frontier change and whether the resulting child lies on the winner lineage (Appendix C.3).

Across tasks, AdaGEPA uses one shared decision point with task-specific evaluator signals to identify the parent's weaknesses and guide proposals. Development search tracks candidate discovery, while search-external evaluation tests the resulting prompts on new examples.

7 Conclusion

Reflective prompt optimization depends on both how a prompt is revised and which feedback examples inform the revision. We proposed AdaGEPA to address the latter choice using task structure and the prompt's observed performance. Our analyses show that this choice matters because improvements on the selected examples do not reliably advance the best score on the full validation set. In our experiments across six tasks, AdaGEPA achieved higher mean validation scores under matched rollout budgets and often found higher-scoring candidates earlier in search. Search-external evaluations on SGD, MultiWOZ, and HoVer showed that these gains can also extend to new examples. These results show that feedback allocation is a distinct design choice in reflective prompt optimization, affecting both when useful prompts are found and whether their gains extend to new examples. Future methods could account for longer-term search value by favoring feedback choices that produce useful parents, even when the immediate candidate does not improve the best score.

AI USE STATEMENT

We used generative AI tools to assist with writing and editing experimental and analysis scripts, organizing and analyzing experiment records, identifying potentially relevant literature, and restructuring, language editing, and polishing of the manuscript. The authors reviewed all AI-assisted outputs, checked reported numerical results against the corresponding experiment records, and verified citations against their original sources. The authors take responsibility for the final text, claims, code, and other artifacts.

REPRODUCIBILITY STATEMENT

Sections 4 and 5.1 describe AdaGEPA's feedback-selection procedure and evaluation design. Appendix A documents the local-to-full proposal-alignment audit. Appendix B specifies the task adapters, data roles and evaluation panels, rollout budgets, model request settings, metrics, and statistical procedures. Appendix C provides additional checkpoint and paired-seed results, controls, and resource accounting. The accompanying anonymous code artifact contains an executable reference implementation of the shared bounded one-slot feedback-allocation contract and its GEPA-compatible batch sampler.

REFERENCES

Lakshya A. Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J. Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective prompt evolution can outperform reinforcement learning, 2025. URL https://arxiv.org/abs/2507.19457.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 41–48, 2009. doi: 10.1145/1553374.1553380.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. MultiWOZ: A large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 5016–5026, 2018. doi: 10.18653/v1/D18-1547.

Xiaoyin Chen, Canwen Xu, Yite Wang, Boyi Liu, Zhewei Yao, and Yuxiong He. Learning to self-evolve, 2026. URL https://arxiv.org/abs/2603.18620.

Jasper Dekoninck, Nikola Jovanović, Tim Gehrunger, Kári Rögnvaldsson, Ivo Petrov, Chenhao Sun, and Martin Vechev. Beyond benchmarks: MathArena as an evaluation platform for mathematics with LLMs, 2026. URL https://arxiv.org/abs/2605.00674.

Ximing Dong, Shaowei Wang, Dayi Lin, and Ahmed E. Hassan. Model performance-guided evaluation data selection for effective prompt optimization. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 2844–2859, 2025. doi: 10.18653/v1/2025.findings-acl.147.

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rockt"aschel. Promptbreeder: Self-referential self-improvement via prompt evolution, 2023. URL https://arxiv.org/abs/2309.16797.

Zhaolin Gao, Yu (Sid) Wang, Bo Liu, Thorsten Joachims, Kianté Brantley, and Wen Sun. $p1$: Better prompt optimization with fewer prompts, 2026. URL https://arxiv.org/abs/2604.08801.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. EvoPrompt: Connecting LLMs with evolutionary algorithms yields powerful prompt optimizers. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2309.08532.

Yichen Jiang, Shikha Bordia, Zheng Zhong, Charles Dognin, Maneesh Singh, and Mohit Bansal. HoVer: A dataset for many-hop fact extraction and claim verification. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 3441–3460, 2020. doi: 10.18653/v1/2020.findings-emnlp.309.

Junbo Li, Boyi Liu, Canwen Xu, Yite Wang, Yuxiong He, Zhangyang Wang, Qiang Liu, and Zhewei Yao. The optimizer is the agent: Reasoning-driven search across prompts, programs, and ML workflows, 2026. URL https://arxiv.org/abs/2608.06714.

Jinming Nian, Zhiyuan Peng, Hongwei Shang, Dae Hoon Park, and Yi Fang. Submodular evaluation subset selection in automatic prompt optimization, 2026. URL https://arxiv.org/abs/2601.03493.

Krista Opsahl-Ong, Michael J. Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. Optimizing instructions and demonstrations for multi-stage language model programs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 9340–9366, 2024. doi: 10.18653/v1/2024.emnlp-main.525.

Young-Joon Park, Seong-Ryeong Lee, Anh-Dung Vo, Minsung Jung, and Daewoo Choi. TwinBandit Prompt Optimizer: Adaptive prompt optimization via synergistic dual MAB-guided feedback. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, pp. 5115–5119, 2025. doi: 10.1145/3746252.3760824.

Reid Pryzant, Dan Iter, Jerry Li, Yin Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with "gradient descent" and beam search. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7957–7968, 2023. doi: 10.18653/v1/2023.emnlp-main.494.

Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hannaneh Hajishirzi. Generalizing verifiable instruction following. In *Advances in Neural Information Processing Systems*, volume 38, pp. 54870–54896, 2025. doi: 10.52202/085713-1645.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 8689–8696, 2020. doi: 10.1609/aaai.v34i05.6394.

Subhash Bangalore Satheesha, Nirvik Pande, Deepthi Duddempudi, and Bharath Dandala. RLMOpt: Adaptive prompt optimization via recursive language models, 2026. URL https://arxiv.org/abs/2608.10471.

Fei Wang, Si Si, Cho-Jui Hsieh, and Inderjit S. Dhillon. APEX: Automated prompt engineering expert with dynamic data selection, 2026. URL https://arxiv.org/abs/2606.11459.

Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Ben Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Sreemanti Dey, Shubh-Agrawal, Sandeep Singh Sandha, Siddartha Naidu, Chinmay Hegde, Yann LeCun, Tom Goldstein, Willie Neiswanger, and Micah Goldblum. LiveBench: A challenging, contamination-limited LLM benchmark. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2406.19314.

Siheng Xiong, Ali Payani, Oguzhan Gungordu, and Faramarz Fekri. DIVE: Unlocking self-improvement in frozen language models through diversity-driven skill evolution, 2026. URL https://arxiv.org/abs/2608.12486.

Hui Xue and Fan Yang. Rethinking self-evolving agents: Do we still need prescribed optimization pipelines?, 2026. URL https://arxiv.org/abs/2608.09629.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2309.03409.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 5824–5836, 2020. URL https://arxiv.org/abs/2001.06782.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. TextGrad: Automatic "differentiation" via text. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2406.07496.

A Local-to-Full Proposal Alignment

We examine local-to-full alignment across 458 proposal transitions from 60 development runs on AIME, LiveBench-Math, and IFBench. Each transition passed the strict local gate and completed full evaluation within the common $B \leq 560$ rollout prefix. The runs include the GEPA default and several historical adaptive policies, so this analysis characterizes proposal-level alignment rather than the effect of a particular method.

For each proposal, the local change is the child-minus-parent score change on the selected minibatch, while the full-set change is the corresponding change on $\mathcal{D}_{\text{pareto}}$. All minibatches have size $b = 3$, so local score sums and averages differ only by a constant factor and have the same sign. Correlations use raw score-sum changes for both quantities. Because the validation-set sizes differ across tasks, the pooled correlations are descriptive. The Pearson and Spearman correlations were 0.076 and 0.108, respectively, indicating weak local-to-full alignment.

Proposal source	Proposals	Positive	Zero	Negative
GEPA default subset	150	85	14	51
All proposals	458	233	54	171

Table 3: Full-validation child-minus-parent outcomes among proposals that passed the local gate and completed full evaluation. The GEPA default row is a subset of the all-proposal row.

Although every proposal passed the local gate, 233 of 458 improved on its parent under full validation. The remaining 225 were unchanged or worse. Together with the low correlations, these outcomes distinguish improvement on the selected minibatch from progress on the full validation set. This analysis follows realized search trajectories rather than forked alternatives and therefore does not estimate the counterfactual advantage of different feedback choices from the same state.

B Implementation and Evaluation Protocol

B.1 Search Setting and Data Roles

All paired comparisons use the same GEPA search implementation, with candidate merging disabled. AdaGEPA intervenes after parent selection and changes only the feedback minibatch used for reflection. Within each comparison, the methods share all remaining search and evaluation settings, including the rollout budget and model request configuration. Search-external evaluations use frozen checkpoint candidates without further prompt optimization.

During optimization, $\mathcal{D}_{\text{feedback}}$ supplies examples and evaluator feedback for reflection and local parent–child comparison, whereas $\mathcal{D}_{\text{pareto}}$ supports full candidate evaluation, parent selection, and final winner selection.

Table 4 summarizes the optimization runs and saved-state sets used in the reported comparisons. Search arms are paired within each comparison and share their initial validation records. Search-external evaluations reuse candidates from the corresponding searches and therefore do not add optimization runs.

B.2 GEPA Search Transition

Before proposal step t, the candidate pool $\mathcal{P}_t$ contains all candidates that have undergone full evaluation on $\mathcal{D}_{\text{pareto}}$. GEPA samples a parent using its Pareto-based selector (Agrawal et al., 2025). The selector prunes redundant candidates from the sampling support and then samples each remaining candidate in proportion to the number of validation instances on which it is tied for the best score. This pruning affects only parent selection and does not remove candidates from $\mathcal{P}_t$.

The default sampler maintains an epoch-shuffled schedule over $\mathcal{D}_{\text{feedback}}$. After parent selection, the next b scheduled instances form the feedback minibatch $\mathcal{M}_t^0$. Running the parent Φ_k on this minibatch produces outputs, scores, execution trajectories, and task-native feedback. The Reflection-LM uses these records to revise one prompt component of the parent and produce a child candidate Φ'.

Table 4: Optimization runs and saved-state sets used in the reported comparisons. For SGD, 5-run and 10-run identify the number of formation searches used to fit the ordering. Same-state comparisons reuse states from their source searches.

Task	Experiment	Repeats
SGD	Ordering formation sets	5 / 10 searches
SGD	Original study (5 formation searches)	5 paired runs
SGD	Extension study (5 formation searches)	5 paired runs
SGD	10-run study (10 formation searches)	10 paired runs
SGD	History-and-coverage and formation controls	5 paired runs
SGD	Random Legal three-arm control	10 paired runs
SGD	Strong-prompt continuation	5 paired continuations
SGD	Same-state comparisons	2×5 source panels
MultiWOZ	Service-aware search and allocation controls	5 paired runs
HoVer	Stage-progress replay	5 paired runs
AIME	Full-minibatch and bounded controls	5 paired runs
LiveBench-Math	Full-minibatch design study	10 paired runs
IFBench	Verifier-allocation comparison	5 paired runs
PUPA	Privacy-feedback reuse	5 paired runs
TACO	Coarse-skill allocation	5 paired runs

The parent and child are evaluated on the same occurrence-level minibatch. The strict local gate advances the child to full validation only if its average minibatch score exceeds that of the parent. The implementation compares the corresponding score sums, which yields the same decision because both candidates are evaluated on the same b occurrences. Under the configuration used in this paper, a child that passes the local gate is fully evaluated on $\mathcal{D}_{\text{pareto}}$ and added to the candidate pool. At the end of search, the candidate with the highest average score on $\mathcal{D}_{\text{pareto}}$ is returned.

B.3 Shared Allocation Contract

The AdaGEPA instantiations for SGD, MultiWOZ, HoVer, IFBench, PUPA, and TACO use the one-slot contract from Section 4. Each begins with a three-occurrence GEPA default minibatch, retains the first two occurrences, and allows a task-native adapter to nominate a replacement for the third. The proposal must satisfy the task's support and schedule rules; otherwise the default minibatch is used. AIME records the earlier full-minibatch allocation controls that motivated this bounded interface.

B.4 Task-Specific Adapters

The shared interface leaves the evidence schema and proposal rule task specific. Table 5 summarizes how each task maps its evaluation evidence to a feedback proposal. The proposal rules and any fitted parameters are fixed before evaluation.

The SGD instantiation uses fitted proposal ordering. Table 6 summarizes its formation signal, frozen models, and runtime selection rule. All fitted quantities remain fixed during evaluation.

Across the one-slot task instantiations, a proposal is applied only when it satisfies the frozen support and schedule rules. If no proposal qualifies, AdaGEPA retains the default minibatch.

B.5 Controls and Same-State Comparisons

Table 7 summarizes the controls used to separate action capacity, proposal support, ordering, and task-specific targeting. The APEX-style and SESS-style controls adapt ideas from their source systems (Wang et al., 2026; Nian et al., 2026). They preserve the surrounding search transition but are not full reproductions of those systems.

Table 5: Task-specific evidence schemas and proposal rules. AdaGEPA uses bounded one-slot proposals, whereas AIME is included as a full-minibatch design study.

Task	Evidence used	Proposal rule	Question examined
SGD	Search progress, service exposure, static task profiles, and selected-parent component deficits.	Frozen models rank legal service-level proposals and, when supported, refine the choice within the selected service.	Component-aware proposal ordering and candidate transfer.
MultiWOZ	Selected-parent component deficits, service exposure, and frozen service metadata.	A fixed service-aware rule prioritizes supported occurrences from services where the selected parent is weak.	Service-deficit targeting versus generic coverage and fitted ordering.
HoVer	Intermediate retrieval progress, unresolved final outcomes, and replay history.	A fixed replay rule prioritizes an eligible occurrence whose trajectory made retrieval progress but did not resolve the task.	Stage-progress-guided feedback reuse.
AIME	Failure history, repeated exposure, and coarse topic clusters.	Full-minibatch design controls prioritize difficult or underrepresented examples rather than applying one-slot replacement.	Whether history and coverage alone improve feedback allocation.
IFBench	Selected-parent verifier deficits and prior exposure under overlapping objectives.	Fixed rules prioritize supported examples associated with weak verifier groups. Alternative variants aggregate these deficits differently.	Feedback allocation under overlapping verifier objectives.
PUPA	The selected prompt component, prior feedback observations, and visible privacy-leakage signals.	A fixed component-aware rule reuses eligible feedback with previously observed privacy leakage while leaving unrelated update types unchanged.	Feedback reuse and candidate retention under coupled objectives.
TACO	Validation-supported skill labels and selected-parent skill deficits.	A fixed rule prioritizes a supported example from an underperforming skill group.	Whether coarse skill identity provides sufficient resolution for feedback allocation.

Table 6: Compact configuration of the fitted SGD proposal ordering.

Component	Specification
Formation evidence	Five designated development searches using the default feedback rule provide observed proposal states and one-step targets.
Ordering target	The target is the observed one-step Balanced-DST frontier gain after a complete proposal transition. Rejected or non-advancing proposals receive zero.
Frozen models	Two ridge models rank supported service-level proposals and optional within-service refinements using search progress, service exposure, static task profiles, and selected-parent deficits.
Runtime selection	The service-level model first identifies a legal proposal. The refinement model may then select a supported occurrence within that service. If neither proposal is available, the adapter retains the default minibatch.

The PUPA component-composition comparator changes candidate components rather than feedback selection and is therefore reported separately.

Table 7: Control designs and the factors they are intended to isolate.

Control	Preserved structure	Changed factor	Purpose
Random Legal	One-slot capacity, legal support, and the search schedule	Samples uniformly from legal replacements without task-aware ordering	Isolate the value of legal replacement alone.
Structured Unordered	AdaGEPA's supported proposal set and one-slot capacity	Removes final occurrence-identity ordering	Separate structured proposal support from fitted ordering.
MultiWOZ coverage	One-slot capacity and prior exposure information	Selects unused examples without parent-derived service deficits	Compare generic coverage with service-deficit targeting.
PUPA coverage	One-slot capacity	Selects unseen examples without observed leakage feedback	Compare generic coverage with observed-feedback reuse.
History and coverage	The surrounding search transition and rollout accounting	Resamples the full minibatch using difficulty, repetition, and coverage	Test the earlier full-minibatch allocation hypothesis.
Formation support	The fitted adapter architecture and evaluation protocol	Varies the number of development searches used to fit proposal ordering	Measure sensitivity to the amount of formation evidence.
APEX-style	The task-matched feedback capacity and search transition	Prioritizes feedback using failure history	Provide a neighboring adaptive mutation-data control.
SESS-style	The search transition and epoch composition	Uses a fixed representative schedule rather than parent-conditioned selection	Compare adaptive allocation with static representativeness.

Table 8: Same-state fork panels. Every selected state supports all actions compared in its panel.

Panel	Source states	Actions compared	Repeats
Four-arm	15 states from five development runs	GEPA default, Random Legal, Structured Unordered, and five-run AdaGEPA	3 per action
Ordering	15 states from another five development runs	Structured Unordered and ten-run AdaGEPA	3 per action

B.5.1 SGD Same-State Fork Protocol

Each audit restores a fixed search state ξ_t and evaluates every compared action using $K = 3$ independent proposal transitions. Randomized controls draw a new legal replacement on each repeat. We estimate immediate frontier gain and its contrast with the default choice a_0 as

$$\widehat{U}(a \mid \xi_t) = \frac{1}{K}\sum_{r=1}^{K}\big(H_{t+1}^{(a,r)} - H_t\big), \qquad \widehat{\mathrm{Adv}}(a \mid \xi_t) = \widehat{U}(a \mid \xi_t) - \widehat{U}(a_0 \mid \xi_t).$$

Repeats are averaged within each state, and state-level effects are then aggregated within the source optimization run.

Results appear in Appendix C.2.3.

B.6 Rollout Budgets and Candidate Checkpoints

The rollout budget B counts logical candidate–example evaluations rather than raw provider requests. A proposal may require both a local parent–child comparison and, if accepted, a full validation pass. Before starting a proposal, the runner checks that the remaining budget can cover the

Table 9: Rollout caps, validation sizes, and key checkpoints used in the reported comparisons.

Task	Rollout cap B	Validation size	Key reported checkpoints
SGD	1120	104	104, 392, 560, 840, 1120
MultiWOZ	1120	104	104, 300, 392, 560, 840, 1120
HoVer	1120	300	300, 624, 1120
AIME	560	45	45, 150, 300, 336, 392, 560
LiveBench-Math	560	50	50, 150, 300, 560
IFBench	560	54	54, 300, 392, 560
PUPA	607	111	111, 405, 607
TACO	1424	256	256, 768, 1424

Table 10: Evaluation settings and task metrics. Search-external panels evaluate frozen checkpoint candidates without further prompt optimization.

Task	Development evaluation	Search-external evaluation	Main metric
SGD	Search validation	In-domain (104); unseen-service (104)	Balanced DST for search; Balanced DST and JGA externally
MultiWOZ	Search validation	Search-external panel (104; official-test split)	Balanced DST; JGA complementary
HoVer	Development search	Search-external panel (253); full test set (300) secondary	Document Exact
AIME	45-example validation split	—	Exact-match accuracy
LiveBench-Math	50-example validation split	—	Correct answers
IFBench	54-example validation set	—	Strict pass rate
PUPA	111-example validation set	—	Privacy–quality score
TACO	256-example validation set	—	Mean problem pass rate

complete transition. A checkpoint includes only candidates whose full validation has finished by that budget.

Rollout caps and reported checkpoints follow the fixed protocol for each task. Because validation-set sizes and proposal schedules differ across tasks, their absolute B values are task-specific. Table 9 lists the rollout cap, validation size, and checkpoints used in the reported comparisons.

Because B is an upper bound, local rejection and full-evaluation reservation can leave the realized number of optimization rollouts below the cap.

B.7 Evaluation Panels, Metrics, and Statistics

Development results use each task's search validation data. Search-external evaluations instead score frozen checkpoint candidates on examples that were not used for optimization, candidate selection, or winner selection. SGD uses two 104-example panels: a train-derived in-domain panel and an official-test panel covering 15 unseen services. MultiWOZ uses a deterministic 104-dialogue panel from its official-test split. HoVer uses a fixed 253-example search-external panel and reports the complete 300-example test set as a secondary result. AIME divides its 90 development examples evenly between feedback and validation.

Both frozen SGD panels report JGA and Balanced DST. For MultiWOZ, Balanced DST is the search metric because it combines joint-state correctness with active intent, requested slots, and slot values, while JGA is reported as a strict complementary measure of complete-state correctness. HoVer's Document Exact score is the fraction of examples for which all three normalized gold supporting-

document titles are retrieved. AIME and LiveBench-Math report exact-match accuracy and also express scores as the number of solved problems. IFBench assigns an item score of one only when the response passes every instruction-specific verifier, and zero otherwise. Its endpoint is the mean of these strict item scores.

For PUPA, let F denote the judge decision that the generated response is at least as good as the reference, and let R denote the reverse comparison. The primary quality component is $q = \mathbf{1}[F \vee (F = R)]$, with $\mathbf{1}[F \wedge \neg R]$ reported as a stricter diagnostic. If ℓ is the fraction of unique private-information entries leaked into the external request, privacy is $p = 1 - \ell$, and the per-example overall score is $(q + p)/2$. Candidate-level quality, privacy, and overall scores are their means over the validation examples. TACO reports mean problem pass rate as its primary endpoint and strict pass rate as a complementary endpoint.

For development protocols evaluated at multiple rollout checkpoints, let $H(B)$ denote the quality of the best budget-complete candidate available by rollout B. Over the evaluated budget range $[B_0, B_1]$, we summarize progress above the initial checkpoint as

$$\text{Gain-AUC} = \frac{1}{B_1 - B_0} \int_{B_0}^{B_1} (H(B) - H(B_0))\, dB.$$

This budget-normalized area rewards both earlier and larger improvements over $H(B_0)$. Gain-AUC is reported in points after scaling by 100.

Optimization seeds are the unit of comparison. For each seed, we first compute the paired difference between AdaGEPA and its reference method, then summarize these differences by their mean, median, W/T/L, and a 95% uncertainty interval. When a frozen candidate has repeated evaluation draws, its scores are averaged before the paired difference is computed. Five-run comparisons use exact seed resampling unless otherwise stated, while PUPA uses a t interval over its five paired differences. Reported aggregates include completed, protocol-conforming runs at the stated checkpoint; failed runs are not replaced.

B.8 Model Requests and Integrity

Table 11: Model request configurations for the reported experiments. T denotes temperature, L the maximum output-token setting, R the configured retry count, and τ the request timeout in seconds. A dash indicates that the field was not explicitly set in the frozen request configuration.

Task	Task-LM or task evaluator	Reflection-LM
SGD, MultiWOZ	`openai/gpt-4.1-mini`; $T = 1$, $L = 1024$, $R = 5$, $\tau = 240$	`openai/gpt-5.1`; $T = 1$, $L = 16000$, $R = 3$, $\tau = 600$
AIME	`openai/gpt-4.1-mini`; $T = 1$, $L = 16000$, $R = 5$, $\tau = 240$	`openai/gpt-5.1`; $T =$ –, $L = 16000$, $R = 3$, $\tau = 600$
IFBench	`openai/gpt-4.1-mini`; $T = 1$, $L = 16000$, $R = 5$, $\tau = 240$	`openai/gpt-5.1`; $T = 1$, $L = 16000$, $R = 3$, $\tau = 600$
TACO	`openai/gpt-4.1-mini`; $T = 1$, $L = 16000$, $R = 3$, $\tau = 600$	`openai/gpt-5.1`; $T = 1$, $L = 16000$, $R = 3$, $\tau = 600$
HoVer	`gpt-4.1-mini`; $T = 1$, $L = 512$, $R = 2$, $\tau = 600$	`gpt-5.1`; $T = 1$, $L = 4096$, $R = 2$, $\tau = 600$
PUPA	`gpt-4.1-mini`; task $T = 1$, $L = 1024$, evaluator $T = 0$, $L = 128$; $R = 5$, $\tau = 180$	`gpt-5.1`; $T = 1$, $L = 4096$, $R = 5$, $\tau = 180$

None of these request configurations explicitly sets `top_p`. The recorded model strings identify the requested endpoints. The protocols do not expose immutable provider-side revision identifiers.

Within each paired comparison, arms share the initial candidate and, where specified, the initial full-validation records. Later Task-LM and Reflection-LM executions are not shared, so batches, proposals, acceptance decisions, and candidate histories may diverge. Formal aggregates include only runs satisfying the frozen protocol, strict budget, budget-complete checkpoint, split boundary, and usage closure. Invalid attempts remain in integrity and technical-resource records but do not enter effect estimates, and frozen comparisons use no replacement seeds. Analyses of cached records use existing outputs without changing search states or scores.

Physical reuse of cached or shared initial-validation work does not remove an arm's protocol-defined rollout charge. Provider retries do not add a task-level rollout, but they enter the physical ledger for

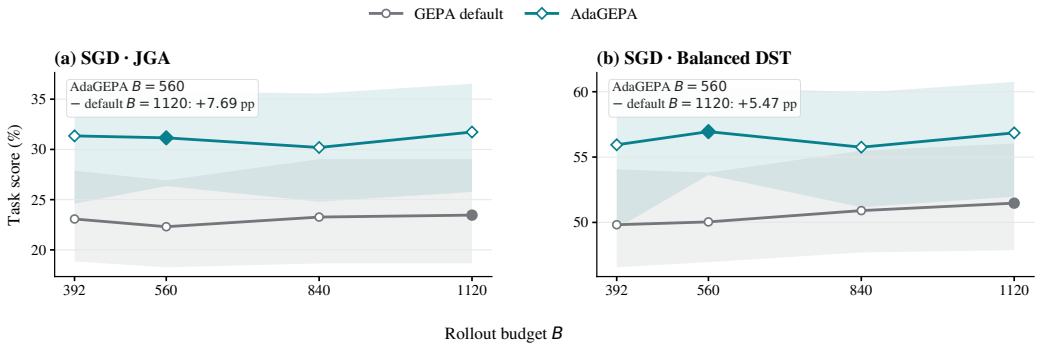

Figure 4: Search-external in-domain SGD quality across frozen rollout checkpoints. Curves show means over five paired optimization runs on the 104-example panel, with 95% seed-resampling intervals. Filled markers identify AdaGEPA at $B = 560$ and the GEPA default at $B = 1120$; annotations report their paired mean differences.

Table 12: Search-external SGD candidate quality across frozen checkpoints on the 104-dialogue panel covering all 15 unseen services. Values are means over five paired optimization runs.

	JGA (%)				Balanced DST (%)			
Arm	$B392$	$B560$	$B840$	$B1120$	$B392$	$B560$	$B840$	$B1120$
GEPA default	27.88	28.08	28.27	28.85	55.81	56.29	56.14	57.33
AdaGEPA	**31.92**	**34.04**	**31.92**	**32.31**	**58.28**	**61.03**	**59.35**	**59.69**

attempts, tokens, configured cost, and wall-clock time. Configured cost is a uniform-rate technical ledger, not a provider invoice. Equal rollout caps need not equalize provider requests, Reflection-LM calls, tokens, configured cost, or wall-clock time.

C Additional Empirical Evidence

Appendix C at a glance.

Search and transfer	Complete checkpoint, seed, and search-external results.
Allocation controls	Comparisons of replacement, task targeting, and ordering.
Search-value retention	Candidate outcomes and task-specific boundaries.
Cross-task synthesis	Task-level outcomes and feedback regimes.
Resource accounting	Physical-resource and formation-cost summaries.

C.1 Search and Transfer Results

C.1.1 SGD 5-Run Ordering: Set A

Figure 4 shows in-domain candidate quality across the frozen checkpoints, while Table 12 reports the corresponding results on the unseen-service panel.

At the matched $B = 560$ checkpoint, AdaGEPA improved mean unseen-service JGA by 5.96 points, with a 95% paired-seed interval of $[4.81, 6.92]$. At $B = 1120$, the mean JGA differences were $+8.27$ points in-domain (W/T/L $= 5/0/0$, interval $[+4.23, +13.65]$) and $+3.46$ points on unseen services (W/T/L $= 4/0/1$, interval $[+0.58, +5.77]$). Comparing AdaGEPA at $B = 560$ with the GEPA default at $B = 1120$ gives $+7.69$ points in-domain and $+5.19$ points on unseen services, with W/T/L $= 5/0/0$ for both comparisons.

C.1.2 Second Five-Run SGD Study

The second five-run SGD study uses the same ordering configuration as the first with a different set of paired optimization runs. Table 13 reports the paired differences behind the arm means in Table 2,

Table 13: Paired SGD differences for the second five-run study. All values are AdaGEPA minus the GEPA default, in points. The search-external results use the existing fixed in-domain and unseen-service panels.

Evaluation	Contrast and metric	Mean	W/T/L	95% paired-seed interval
Development	Gain-AUC, Balanced DST	$+0.73$	3/0/2	$[-1.06, +2.52]$
Development	$B = 560$ vs. $B = 560$, Balanced DST	$+1.79$	4/0/1	$[+0.03, +4.00]$
Development	$B = 1120$ vs. $B = 1120$, Balanced DST	-0.40	3/0/2	$[-1.67, +0.87]$
In-domain panel	$B = 560$ vs. $B = 560$, JGA	$+4.04$	3/0/2	$[-2.12, +11.54]$
In-domain panel	$B = 1120$ vs. $B = 1120$, JGA	$+1.54$	2/1/2	$[-1.15, +4.23]$
In-domain panel	$B = 560$ vs. default $B = 1120$, JGA	-0.19	2/0/3	$[-3.65, +3.27]$
Unseen-service panel	$B = 560$ vs. $B = 560$, JGA	$+5.71$	5/0/0	$[+3.72, +7.69]$
Unseen-service panel	$B = 1120$ vs. $B = 1120$, JGA	$+0.77$	3/0/2	$[-5.00, +6.54]$
Unseen-service panel	$B = 560$ vs. default $B = 1120$, JGA	$+0.58$	3/0/2	$[-3.91, +5.00]$

together with Gain-AUC and cross-budget comparisons. In the cross-budget rows, AdaGEPA at $B = 560$ is compared with the GEPA default at $B = 1120$.

At $B = 560$, the mean differences favor AdaGEPA on development and on both search-external panels. The differences narrow by $B = 1120$ as the default search continues to improve.

C.1.3 Ten-Run SGD Study

The ten-run SGD study comprises ten paired optimization searches under a fixed AdaGEPA configuration. Table 14 reports paired development and search-external differences at the earlier checkpoint and final endpoint, together with the corresponding cross-budget comparisons. Development uses Balanced DST and Gain-AUC; the search-external rows use JGA or Balanced DST as labeled. Differences and intervals are in points.

The mean matched-budget difference at $B = 560$ favors AdaGEPA on development, in-domain, and unseen-service evaluations. At $B = 1120$, the development and unseen-service means are close to the default, while the in-domain JGA difference remains positive on average.

Table 15 reports the per-run unseen-service Balanced DST and JGA differences for the three SGD studies at $B = 560$ and $B = 1120$.

C.1.4 SGD Formation-Prompt Comparisons

Direct deployment. Five GEPA default formation searches produced a development-selected prompt that could be deployed without further optimization. We compared this prompt with AdaGEPA's $B = 560$ candidates on the same unseen-service panel (Table 16).

The formation prompt achieved 36.54% JGA, compared with an AdaGEPA mean of 32.69%. Formation search can therefore provide a strong prompt before subsequent optimization.

Continuation from the formation prompt. To examine whether adaptive feedback selection adds value from a stronger initialization, the GEPA default and AdaGEPA started from the same development-selected formation prompt. By $B = 1120$, they had improved on that prompt by 3.22 and 3.00 Balanced DST points, respectively, leaving AdaGEPA 0.22 points below the default. Both methods improved the stronger starting prompt, but their mean gains were similar.

C.1.5 MultiWOZ Search-External Checkpoint Evaluation

Frozen candidates saved at $B = 560$ and $B = 1120$ in five paired MultiWOZ searches were evaluated on a fixed 104-dialogue search-external panel drawn from the official-test split. Table 17 reports the per-run differences at both checkpoints.

Table 14: Paired SGD differences for the ten-run ordering study. In-domain results are from a later frozen-candidate evaluation.

Evaluation	Contrast and metric	Mean	Median	W/T/L	95% paired-seed interval
Development	Gain-AUC	$+0.99$	$+0.11$	5/0/5	$[-2.46, +4.72]$
Development	$B = 560$ vs. $B = 560$, Balanced DST	$+1.58$	$+0.51$	6/0/4	$[-2.15, +5.86]$
Development	$B = 1120$ vs. $B = 1120$, Balanced DST	-0.03	-0.24	5/0/5	$[-3.02, +2.90]$
Development	$B = 560$ vs. default $B = 1120$, Balanced DST	-1.61	-1.48	4/0/6	$[-4.38, +1.12]$
In-domain panel	$B = 560$ vs. $B = 560$, JGA	$+5.58$	$+5.29$	7/0/3	$[-0.38, +11.44]$
In-domain panel	$B = 1120$ vs. $B = 1120$, JGA	$+2.21$	$+2.88$	7/0/3	$[-1.92, +5.87]$
In-domain panel	$B = 560$ vs. default $B = 1120$, JGA	$+1.25$	$+4.81$	6/0/4	$[-3.94, +6.06]$
Unseen-service panel	$B = 560$ vs. $B = 560$, JGA	$+2.02$	$+1.92$	6/0/4	$[-2.50, +6.73]$
Unseen-service panel	$B = 1120$ vs. $B = 1120$, JGA	-0.38	$+0.48$	5/0/5	$[-3.65, +2.50]$
Unseen-service panel	$B = 560$ vs. default $B = 1120$, JGA	-1.35	0.00	5/0/5	$[-5.48, +2.60]$
Unseen-service panel	$B = 560$ vs. $B = 560$, Balanced DST	$+1.11$	$+1.90$	6/0/4	$[-2.86, +5.11]$
Unseen-service panel	$B = 1120$ vs. $B = 1120$, Balanced DST	-0.24	$+0.35$	5/0/5	$[-3.23, +2.46]$
Unseen-service panel	$B = 560$ vs. default $B = 1120$, Balanced DST	-1.37	-0.14	5/0/5	$[-4.98, +2.07]$

Table 15: Per-run SGD search-external differences on the unseen-service panel. Values are AdaGEPA minus the GEPA default in percentage points.

		Balanced DST		JGA	
SGD study	**Run**	$B = 560$	$B = 1120$	$B = 560$	$B = 1120$
Original	1	$+2.96$	-0.38	$+3.85$	-1.92
	2	$+5.19$	$+3.46$	$+5.77$	$+5.77$
	3	$+5.15$	$+1.05$	$+5.77$	$+2.88$
	4	$+4.99$	$+5.46$	$+6.73$	$+6.73$
	5	$+5.42$	$+2.23$	$+7.69$	$+3.85$
Extension	1	$+1.07$	$+2.81$	$+3.53$	$+3.85$
	2	$+5.10$	-4.58	$+7.05$	-7.69
	3	$+3.92$	$+6.27$	$+5.77$	$+11.54$
	4	$+6.89$	-3.12	$+9.29$	-5.77
	5	$+1.37$	$+1.12$	$+2.88$	$+1.92$
10-run	1	$+5.10$	$+2.49$	$+8.65$	$+3.85$
	2	$+0.14$	-4.03	$+0.96$	-3.85
	3	$+7.74$	$+4.52$	$+9.62$	$+3.85$
	4	-6.01	-1.49	-6.73	-3.85
	5	-5.42	-1.40	-5.77	-0.96
	6	$+3.66$	$+6.27$	$+2.88$	$+5.77$
	7	-8.13	-10.32	-6.73	-11.54
	8	$+11.02$	$+2.10$	$+13.46$	$+1.92$
	9	-4.53	$+2.68$	-5.77	$+3.85$
	10	$+7.53$	-3.28	$+9.62$	-2.88

At $B = 560$, the mean and median differences were $+1.56$ and $+3.30$ Balanced DST points (W/T/L $= 3/0/2$, interval $[-2.74, +5.85]$), and 0.00 and $+2.88$ JGA points (W/T/L $= 3/0/2$, interval $[-6.73, +6.73]$). At $B = 1120$, the corresponding mean and median differences were $+3.12$ and

Table 16: Direct deployment comparison on unseen-service SGD. Scores are JGA percentages, and the difference is AdaGEPA minus the formation prompt in percentage points.

Evaluation	**Formation prompt**	**AdaGEPA** $B = 560$	**Difference**	**95% interval**
Unseen-service JGA	36.54	32.69	-3.85	$[-13.97, +5.82]$

Table 17: MultiWOZ frozen-candidate evaluation on the fixed 104-dialogue search-external panel drawn from the official-test split. Values are AdaGEPA minus the GEPA default in percentage points.

	Balanced DST		**JGA**	
Run	$B = 560$	$B = 1120$	$B = 560$	$B = 1120$
1	$+3.30$	$+3.49$	$+2.88$	$+3.85$
2	-4.48	$+3.05$	-9.62	-1.92
3	$+6.11$	$+3.67$	$+8.65$	$+4.81$
4	$+6.86$	$+10.01$	$+6.73$	$+17.31$
5	-4.01	-4.64	-8.65	-9.62

$+3.49$ Balanced DST points (W/T/L $= 4/0/1$, interval $[-1.38, +7.32]$), and $+2.88$ and $+3.85$ JGA points (W/T/L $= 3/0/2$, interval $[-4.23, +10.96]$).

C.1.6 HoVer Search-External Evaluation

Frozen candidates saved at $B = 624$ and $B = 1120$ in five paired HoVer searches were evaluated on the same fixed 253-example search-external panel. Table 18 reports the per-run Document Exact differences at both checkpoints.

At $B = 624$, the mean and median differences were $+1.50$ and 0.00 points (W/T/L $= 2/1/2$, interval $[-5.53, +10.43]$). At $B = 1120$, they were $+1.90$ and $+1.19$ points (W/T/L $= 3/0/2$, interval $[-3.24, +8.46]$). On the complete 300-example test set at $B = 1120$, the corresponding mean difference was $+1.53$ points.

C.2 What Drives Feedback Allocation?

C.2.1 SGD: Full-Minibatch History and Coverage

The history-and-coverage sampler ranks feedback examples using loss gaps, repeated failures, and coarse service coverage, then resamples all three minibatch positions. Table 19 compares it with the GEPA default over five matched SGD development runs.

The sampler led the default by 1.29 Balanced DST points at $B = 392$, but the two arms were nearly tied at $B = 560$, and the default continued to improve at later checkpoints. Across the five runs, 148 of 150 selected slots were first-time occurrences, so the rule operated mainly as unseen-example prioritization. Its component scores moved in different directions: active intent improved by 1.15 points, while requested-slot, slot-value, and joint exact scores decreased by 4.94, 1.02, and 3.08 points, respectively. This pattern motivated a narrower intervention that preserves most of the scheduled minibatch and directs one position using task-specific evidence.

C.2.2 SGD: One-Slot Replacement and Ordering

The SGD controls address three questions: whether legal replacement alone is sufficient, whether task-structured proposals and fitted ordering add value, and how performance changes with the amount of formation evidence. Table 20 reports paired differences within each matched comparison. Rows use their respective paired run sets and are not pooled across studies.

Random Legal remained close to the GEPA default, indicating that one-slot replacement alone did not account for the improvement. AdaGEPA achieved higher mean Gain-AUC and endpoint quality than Random Legal. Structured Unordered recovered most of AdaGEPA's mean improvement, while fitted ordering provided a smaller additional gain in this configuration.

Table 18: Per-run HoVer search-external Document Exact differences. Values are AdaGEPA minus the GEPA default on the fixed 253-example panel, in percentage points.

Run	$B = 624$	$B = 1120$
1	$+1.98$	-0.79
2	-9.88	$+1.19$
3	-1.98	-6.32
4	$+17.39$	$+13.83$
5	0.00	$+1.58$

Table 19: Matched SGD development results for the full-minibatch history-and-coverage sampler. Checkpoints report Balanced DST percentages, and Gain-AUC is reported in points.

Arm	$B392$	$B560$	$B840$	$B1120$	**Gain-AUC**
GEPA default	51.12	53.87	55.95	56.30	9.44
History-and-coverage	52.41	53.80	53.80	53.96	8.99

Service-level targeting and component refinement. The service-level-only control targets weak services but omits within-service refinement based on the parent's intent and slot deficits. Table 21 compares its frozen candidates with the other arms from the first five-run SGD study on the unseen-service panel.

AdaGEPA exceeded the service-level-only control by 4.81 JGA points at $B = 560$ and 0.38 points at $B = 1120$.

The remaining controls examine alternative guidance rules and formation support. APEX-style prioritizes examples using failure history, whereas SESS-style places a fixed representative subset at the beginning of each epoch. Formation-size comparisons were nonmonotonic. The ten-run configuration had the strongest mean results among these variants, the fifteen-run configuration retained a positive Gain-AUC difference but little endpoint gain, and the twenty-run configuration did not improve either summary. Full paired results appear in Table 20.

C.2.3 SGD: Same-State Allocation Comparisons

The comparison protocol and state selection are specified in Appendix B.5.1. The reference panel compares four feedback choices from the same saved SGD states.

Table 22 reports frontier gain after averaging the three transition repeats within each state and then aggregating by source seed. Random Legal had the largest positive mean difference over the GEPA default. Structured Unordered was close to Random Legal, while AdaGEPA did not add mean frontier gain over Structured Unordered on this panel. Three of the 45 AdaGEPA and Structured Unordered references selected the same action and reused the corresponding physical transition.

Ten-run configuration. A second panel compared the ten-run AdaGEPA ordering with its corresponding Structured Unordered ablation. Both arms made 45 realized one-slot replacements. Their local-pass counts were similar, at 41 for AdaGEPA and 42 for Structured Unordered, while 11 and 4 transitions, respectively, advanced the pre-existing frontier.

The paired mean difference was $+0.533$ percentage points, with median $+0.281$, W/T/L $= 3/1/1$, and a 95% paired-seed interval of $[-0.001, +1.135]$ points. Thus, with replacement frequency and the structured pool held fixed, the ten-run ordering provided a positive mean increment on this panel. The effect varied across states and was concentrated most strongly in two of the five source runs. One of the 45 references selected the same action in both arms and reused the corresponding physical transition. The reference five-run and ten-run audits use different state panels, so together they characterize configuration-dependent ordering value rather than a matched formation-size effect.

Table 20: Paired SGD development controls. Changes are first arm minus comparator. Gain-AUC differences are points and endpoint differences are percentage points. Each row uses its own matched run set.

Contrast	**Gain-AUC / W/T/L**	**Endpoint / W/T/L**
One-slot replacement and ordering		
AdaGEPA − default	$+3.97$ / $5/0/0$	$+3.92$ / $5/0/0$
Structured Unordered − default	$+3.24$ / $3/0/2$	$+3.36$ / $3/0/2$
AdaGEPA − Structured Unordered	$+0.74$ / $4/0/1$	$+0.56$ / $3/0/2$
AdaGEPA − Random Legal	$+1.02$ / $5/0/5$	$+1.12$ / $5/0/5$
Random Legal − default	$+0.03$ / $5/0/5$	-0.83 / $5/0/5$
Full-minibatch baseline and formation-size sensitivity		
AdaGEPA (5 formation runs) − history-and-coverage	$+0.27$ / $3/0/2$	$+1.90$ / $4/0/1$
AdaGEPA (10 formation runs) − history-and-coverage	$+2.86$ / $3/0/2$	$+4.03$ / $3/0/2$
10-run formation − 5-run formation	$+2.59$ / $4/0/1$	$+2.13$ / $4/0/1$
AdaGEPA (10 formation runs) − default	$+2.41$ / $3/0/2$	$+1.69$ / $3/0/2$
AdaGEPA (15 formation runs) − default	$+1.60$ / $5/0/0$	$+0.48$ / $4/0/1$
AdaGEPA (20 formation runs) − default	-0.18 / $3/0/2$	-0.99 / $2/0/3$
Neighboring feedback-allocation controls		
APEX-style failure prioritization − default	$+1.01$ / $3/0/2$	$+1.92$ / $3/0/2$
SESS-style representative schedule − default	$+2.17$ / $4/0/1$	$+2.73$ / $3/0/2$

Table 21: Unseen-service JGA (%) for the first five-run SGD study. The service-level-only control omits AdaGEPA's within-service component refinement.

Arm	$B = 560$	$B = 1120$
GEPA default	28.08	28.85
Service-level-only	29.23	31.92
AdaGEPA	34.04	32.31

C.2.4 MULTIWOZ: SERVICE TARGETING AND ORDERING

Table 24 separates two choices in the MultiWOZ adapter: whether replacement responds to service-level deficits in the parent prompt, and whether supported occurrences within the targeted service are ordered by a fitted model. Coverage-only samples uniformly from unused feedback occurrences without using service information. The fitted ordering was trained on MultiWOZ default-search traces.

Service-aware AdaGEPA exceeded the GEPA default by 1.79 endpoint points and 1.48 Gain-AUC points, with four of five paired runs favoring it on both measures. Relative to Coverage-only, the corresponding differences were $+2.15$ and $+1.55$ points, with paired intervals of $[0.59, 3.48]$ and $[0.08, 3.09]$. Coverage-only remained close to the GEPA default. Adding fitted within-service ordering reduced the endpoint by 2.83 points relative to service-aware AdaGEPA, with all five paired runs favoring the latter. In these runs, the improvement beyond coverage-only came from service-deficit targeting, while fitted ordering provided no further gain.

C.2.5 AIME: FULL-MINIBATCH AND ONE-SLOT CONTROLS

Relative to the GEPA default, V1 changed Gain-AUC by -1.8738 points and the $B = 560$ endpoint by $-1.6/45$, both with W/T/L $= 2/0/3$. The APEX-style control changed these outcomes by -1.7837 points and $-1.8/45$, with W/T/L $= 2/0/3$ and $1/1/3$, respectively (Table 25).

A bounded one-slot diagnostic on the same runs limited the coarse cluster-aware V1 rule to one replacement while retaining two examples from the default minibatch. At $B = 560$, this bounded variant recovered 1.2 correct answers relative to full-minibatch V1, reducing its endpoint gap from 1.6 to 0.4 answers below the GEPA default. Its Gain-AUC improved by 0.9845 points over full-minibatch V1 but remained 0.8893 points below the GEPA default. Limiting the intervention there-

Table 22: Reference-configuration same-state contrasts on saved SGD development states. Frontier-gain differences and intervals are percentage points. Each seed contributes three states, with three transition repeats per arm and state.

Contrast	Mean	Median	W/T/L	95% paired-seed interval
AdaGEPA − Structured Unordered	-0.187	-0.191	2/0/3	$[-0.454, +0.053]$
Structured Unordered − Random Legal	-0.007	-0.031	2/0/3	$[-0.641, +0.749]$
Random Legal − default	$+0.312$	$+0.301$	4/0/1	$[-0.022, +0.735]$
AdaGEPA − default	$+0.118$	$+0.039$	3/0/2	$[-0.573, +0.897]$

Table 23: Same-state frontier gains for the ten-run AdaGEPA configuration. Values are percentage points after averaging three transition repeats within each state.

Source run	AdaGEPA	Structured Unordered	Difference
1	0.000	0.144	-0.144
2	0.000	0.000	0.000
3	1.739	0.167	$+1.572$
4	0.413	0.132	$+0.281$
5	0.958	0.000	$+0.958$
Mean	0.622	0.089	$+0.533$

fore reduced the endpoint loss, while the V1 selection direction still did not improve the full search curve over the GEPA default.

C.2.6 IFBench: Verifier Aggregation

The IFBench development comparison tested two ways to aggregate overlapping verifier feedback. Mean-Beta used average verifier deficits, whereas All-pass-Beta scored joint verifier failure risk. Table 26 reports their search outcomes and Beta-branch actions.

Mean-Beta exceeded the GEPA default by 2.96 endpoint points with W/T/L $= 3/2/0$ and by 1.23 Gain-AUC points with W/T/L $= 4/1/0$. All-pass-Beta exceeded the GEPA default by 0.74 endpoint points despite executing substantially more replacements. The contrast separates the direction supplied by average verifier deficits from intervention frequency alone.

C.2.7 PUPA: Privacy Feedback Reuse

Observed-Deficit Replay (ODR) reuses a previously observed example with positive privacy leakage. The PUPA development comparison uses five matched optimization runs and includes the GEPA default and a one-slot coverage control. Relative to the coverage control, replay changed the endpoint by $+0.006867$ with W/T/L $= 3/0/2$ and Gain-AUC by $+0.004849$ with W/T/L $= 4/0/1$. The corresponding t intervals were $[-0.040123, +0.053856]$ and $[-0.020247, +0.029945]$. Relative to the GEPA default, the endpoint changed by $+0.012602$ and Gain-AUC by $+0.001410$, both with W/T/L $= 3/0/2$ and intervals crossing zero. Replay made 24 privacy-facing decisions and realized 15 one-slot replacements. All $15/15$ reused a previously observed occurrence with positive privacy leakage, whereas the coverage control reused none.

An earlier one-slot coverage rule replaced the third example in all 29 privacy-facing decisions, but none of these replacements reused previously observed privacy feedback. Its descriptive mean endpoint difference from the GEPA default was $+0.02$ percentage points; privacy and quality moved in opposite directions in four of five seeds. A separate earlier privacy-conditioned sampler could replace the entire three-example minibatch. Relative to the GEPA default, privacy rose by 1.83 points and quality fell by 3.06 points, leaving the overall score 0.61 points lower. These historical controls diagnose different intervention capacities; they were not designed as a paired one-slot versus full-minibatch ablation.

Table 24: MultiWOZ component controls over five paired development searches. Endpoint values are Balanced DST percentages, and Gain-AUC is reported in points. Coverage-only removes service targeting, while fitted ordering ranks supported occurrences within the service-aware proposal space.

Arm	Endpoint	Gain-AUC
GEPA default	53.59	11.46
Coverage-only	53.24	11.39
Service-aware AdaGEPA	**55.39**	**12.94**
Service-aware + fitted ordering	52.56	11.68

Table 25: AIME development results over five matched optimization runs. Checkpoint scores are mean correct answers out of 45 validation problems; Gain-AUC is in points. V1 and APEX-style both resample the full three-example feedback minibatch.

Method	Gain-AUC	$B = 150$	$B = 300$	$B = 336$	$B = 392$	$B = 560$
GEPA default	7.33	18.4	20.2	20.2	20.2	20.4
V1	5.45	16.2	17.4	18.0	18.4	18.8
APEX-style	5.54	16.8	18.2	18.2	18.2	18.6

C.3 Where Allocation Value Persists or Disappears

The mechanism funnels separate action realization from downstream search value. Nomination, support, and action apply only to adaptive arms. After full evaluation, parent improvement means that a candidate outperforms its parent on the full validation set. Frontier advancement means that it raises the best validation score found so far. Winner-lineage membership means that it lies on the parent-child path to the final winner, including the winner itself. These three outcomes are complementary rather than consecutive stages. The counts below describe realized search trajectories; optimization seed remains the unit used for statistical comparison.

For adaptive arms, downstream counts follow selected adaptive choices. The decision count also includes steps at which no adaptive choice was selected. HoVer's logs do not distinguish nomination from support. The extended HoVer audit reports marginal outcomes without intersection counts. Figure 5 uses decisions as the denominator for action selection and selected choices for all downstream rates.

The counts locate different points of attrition. SGD and MultiWOZ executed nearly every nominated replacement, so their outcomes cannot be explained by action inactivity. In MultiWOZ, the service-aware adapter and the GEPA default produced the same number of parent improvements, while the adapter retained slightly more frontier and lineage candidates. PUPA lost opportunities first at support and then at candidate formation, despite successfully reusing observed privacy feedback. Seven replay-induced candidates were accepted, and three appeared in final winner lineages.

AIME full-minibatch controls. On AIME, both history-and-coverage V1 (the coarse-cluster control) and APEX-style failure prioritization resample the full minibatch. Their action capacity is therefore reported separately from the one-slot adapters in Figure 5. The two controls made 55 and 53 adaptive choices and produced 46 and 47 fully evaluated candidates, respectively. These candidates produced 23 and 26 parent improvements, 13 and 10 frontier advances, and 10 and 12 winner-lineage memberships. The GEPA default produced 46 fully evaluated candidates, 18 parent improvements, 10 frontier advances, and 11 winner-lineage memberships. These counts do not by themselves determine cumulative search quality. The coarse-cluster control advanced the frontier 13 times, but those advances summed to 36 validation points and occurred later on the budget axis, compared with 44 points over 10 advances for the GEPA default. APEX-style produced 26 parent improvements versus 18 for the GEPA default, yet the positive parent-relative gains summed to 75 points for both arms. The magnitude and timing of realized gains therefore reconcile the higher event counts with the lower Gain-AUC and endpoint scores of both controls.

The following task-specific analyses examine replay, component composition, and feedback-pool saturation.

Table 26: IFBench three-arm development comparison over five matched optimization runs. Scores are strict all-verifier pass percentages. Beta choices count only the final Beta branch, whereas the mechanism funnel also counts support-building Alpha replacements. Full eval. and frontier here refer to Beta-induced children.

Arm	$B = 300$	$B = 392$	$B = 560$	**Gain-AUC**	**Beta choices**	**Full eval.**	**Frontier**
GEPA default	47.41	47.78	48.89	2.92	–	–	–
Mean-Beta	47.41	47.41	**51.85**	**4.15**	8	2	2
All-pass-Beta	47.78	49.26	49.63	3.81	39	5	1

Table 27: PUPA development endpoints over five matched optimization runs. Scores and differences are overall privacy–quality percentages and percentage points. The coverage control uses the same one-slot capacity without reusing observed leakage feedback.

Run	**GEPA default**	**Coverage control**	**ODR**	**ODR − default**
1	80.80	81.13	85.10	$+4.30$
2	86.94	86.75	84.59	-2.34
3	85.10	85.14	86.64	$+1.54$
4	83.64	86.40	90.75	$+7.11$
5	86.70	86.64	82.39	-4.31
Mean	84.64	85.21	85.90	$+1.26$

C.3.1 HoVer Stage-Progress Replay

Bounded stage-progress replay was evaluated against the GEPA default on five new HoVer development seeds. It averaged $+0.024667$ over the GEPA default with W/T/L $= 3/1/1$, median $+0.003333$, and interval $[-0.006000, +0.069333]$. The replay arm used an average of 14.4 more metric calls and 6.98% more configured cost. A post-run lineage audit found an intervention-derived winner in two of five seeds. The corresponding frozen-candidate results on search-external test examples appear in Appendix C.1.6.

C.3.2 PUPA Component-Composition Comparator

The component-composition comparator averaged -0.001310 against the GEPA default with W/T/L $= 2/0/3$. Composition support appeared in two of five seeds, once only at the terminal budget, while an action was selected and evaluated in one seed. This comparator changes candidate components and lies outside the feedback-occurrence intervention.

C.3.3 LiveBench-Math: Feedback-Pool Saturation

LiveBench-Math provides a small-pool view of the full-minibatch design study. The protocol uses 50 feedback examples, 50 development-validation examples, and a rollout cap of $B = 560$. The cluster-aware V1 sampler combines loss gap, repetition control, and coarse mathematical-topic coverage. It can resample all three examples in a minibatch, so it predates AdaGEPA's one-slot contract.

Table 31 reports the combined results over ten paired development runs. The cluster-aware sampler led the GEPA default by 0.4, 0.8, and 0.5 correct answers at $B = 150$, 300, and 560, respectively, and its mean Gain-AUC was 1.397 question-steps higher. By $B = 560$, the default schedule had covered 99.0% of the feedback pool on average. The narrowing gap is consistent with allocation having less influence once the default schedule has exposed reflection to nearly all available examples.

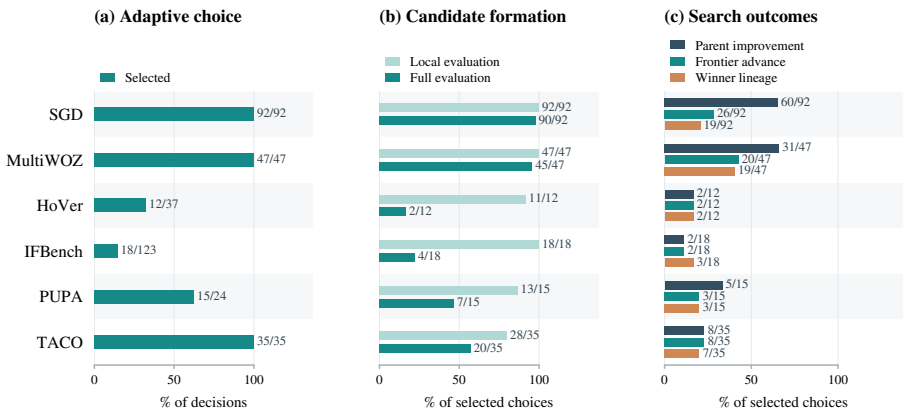

Figure 5: Feedback selection and candidate outcomes across six tasks. (a) Adaptive choices among proposal decisions. (b) Local and full evaluations following selected choices. (c) Parent improvements, frontier advances, and winner-lineage membership. Bars show percentages, with counts and denominators at their ends. Panels (b,c) use selected choices as the denominator. Counts are pooled across seeds within each task, and outcomes in (c) may overlap. IFBench uses the Mean-Beta development comparison, and TACO uses its coarse-skill runs.

Table 28: Action realization and candidate formation in the reported search comparisons. A dash indicates that an adaptive stage does not apply or cannot be uniquely reconstructed from the available records. Counts are stage-specific.

Task	Arm	Dec.	Nom.	Sup.	Act	Refl.	Local eval	Local pass	Full eval
SGD	AdaGEPA	92	92	92	92	92	92	90	90
SGD	Random Legal	95	95	95	95	95	95	90	90
SGD	default	92	–	–	–	92	92	90	90
MultiWOZ	default	47	–	–	–	46	46	45	45
MultiWOZ	Service-aware adapter	47	47	47	47	47	47	45	45
HoVer	default	24	–	–	–	23	23	10	10
HoVer	Stage-progress replay	37	–	–	12	11	11	2	2
IFBench	default	141	–	–	–	135	135	28	28
IFBench	Mean-Beta allocation	123	18	18	18	18	18	4	4
PUPA	default	23	–	–	–	19	17	12	12
PUPA	Coverage control	23	23	22	22	–	17	13	13
PUPA	Observed-deficit replay	24	24	15	15	–	13	7	7
TACO	default	43	–	–	–	36	36	20	20
TACO	Coarse-skill allocation	35	35	35	35	28	28	20	20

Table 29: Outcomes of fully evaluated candidates in the same comparisons. Frontier advancement and winner-lineage membership are complementary outcomes, and their intersection counts candidates satisfying both.

Task	Arm	Full eval	Parent gain	Frontier	Lineage	Both
SGD	AdaGEPA	90	60	26	19	17
SGD	Random Legal	90	60	30	22	18
SGD	default	90	66	32	25	21
MultiWOZ	default	45	31	19	16	12
MultiWOZ	Service-aware adapter	45	31	20	19	17
HoVer	default	10	9	9	6	–
HoVer	Stage-progress replay	2	2	2	2	–
IFBench	default	28	10	6	8	4
IFBench	Mean-Beta allocation	4	2	2	3	2
PUPA	default	12	11	8	6	6
PUPA	Coverage control	13	9	7	5	4
PUPA	Observed-deficit replay	7	5	3	3	3
TACO	default	20	9	6	5	4
TACO	Coarse-skill allocation	20	8	8	7	6

Table 30: Paired HoVer development results. Scores and differences are percentages and percentage points, respectively. Replays count executed stage-progress replacements.

Run	GEPA default	AdaGEPA	Difference	Replays
1	43.00	43.00	0.00	1
2	43.33	43.67	+0.33	2
3	43.67	42.00	−1.67	2
4	38.33	49.67	+11.33	4
5	44.00	46.33	+2.33	3
Mean	42.47	44.93	+2.47	2.4

Table 31: LiveBench-Math development results over ten paired runs. Scores count correct answers out of 50 validation examples. Cluster is the historical full-minibatch V1 sampler, not the one-slot AdaGEPA method.

Method	$B = 150$	$B = 300$	$B = 560$	Gain-AUC
GEPA default	37.6	39.4	40.7	4.877
Cluster V1	38.0	40.2	41.2	6.274

C.4 CROSS-TASK SYNTHESIS

Table 32: Task-level outcome summary. Differences are relative to the GEPA default in each task's native metric. Evaluation setting and intervention capacity are shown explicitly, and outcomes are not averaged across tasks.

Task / capacity	**Evaluation setting**	**Native endpoint**	**Principal outcome relative to the GEPA default**
SGD / one-slot	Frozen in-domain and unseen-service panels; three ordering configurations	JGA	At $B = 1120$, in-domain differences were $+8.27$, $+1.54$, and $+2.21$ points across the original, extension, and 10-run studies; unseen-service differences were $+3.46$, $+0.77$, and -0.38 points.
MultiWOZ / one-slot	Development search; frozen official-test panel	Balanced DST	Development endpoint $+1.79$ points; search-external $+3.12$ at $B = 1120$.
HoVer / one-slot	Development search and frozen search-external panel	Document Exact	Development endpoint $+2.47$ points and search-external endpoint $+1.90$ points at $B = 1120$.
AIME / full and one-slot V1	Five-seed development diagnostic	Correct answers out of 45	Full-minibatch V1 trailed the GEPA default by 1.6 correct answers. Bounded one-slot V1 reduced this gap to 0.4 answers; its Gain-AUC remained 0.89 points lower.
LiveBench-Math / full-minibatch V1	Ten-seed development search	Correct answers out of 50	Cluster-aware V1 led by 0.5 correct answers at $B = 560$, compared with 0.8 at $B = 300$.
IFBench / one-slot	Development comparison	Truncation-robust strict endpoint	Mean-Beta $+2.96$ percentage points in the reported five-seed development comparison.
PUPA / one-slot	New-seed development comparison	Privacy–quality score	Observed-deficit replay $+1.26$ percentage points.
TACO / one-slot	Formation-stage development comparison	Mean-pass / strict pass	$+0.72/+1.09$ percentage points; W/T/L $= 3/0/2$ for both.

C.4.1 Feedback Geometry Across Tasks

Table 33: A hypothesis-generating synthesis of observed feedback-allocation regimes, organized by directional structure, search opportunity, and closed-loop retention.

Task	Directional structure	Search opportunity	Closed-loop retention	Evidence outcome
SGD	Hierarchical service–component alignment	Strongest opportunity before the GEPA default's later catch-up	Structured replacement improves matched-budget early discovery, while cross-budget transfer varies across configurations	Original SGD transfer result and matched-budget early signal
MultiWOZ	Parent-specific service deficits beyond generic unseen coverage	The development lead appears early and remains positive at full budget	More frontier advances and winner-lineage candidates than the GEPA default	Positive mean development and search-external differences across five paired runs
HoVer	Intermediate retrieval-stage progress	Replay opportunities arise after partial progress	Some replay-induced updates reach winner lineages, with retention depending on the trajectory	Positive mean development and search-external differences across five paired runs
AIME	Coarse clusters and failure history	Limited headroom against the GEPA default's strong frontier	Additional parent improvements do not accumulate as frontier gains	Low-structure, default-strong boundary
LiveBench-Math	Coarse mathematical-topic clusters and loss gaps	The small feedback pool nears full coverage by $B = 560$	The mean lead narrows from $0.8/50$ at $B = 300$ to $0.5/50$ at $B = 560$	Full-minibatch V1 precursor with diminishing allocation headroom
IFBench	Overlapping verifier deficits	Frontier-advancing events are rare	Shared prompt updates redistribute gains across verifier objectives	Pre-decision observability and cross-objective interference boundary
PUPA	Observed privacy leakage supplies a replay direction	Fifteen of 24 privacy-facing decisions realize a supported one-slot replay	Seven replay-induced candidates pass the local gate and three reach winner lineages	Weak positive development signal with a retention boundary
TACO	Validation-supported primary-skill deficits	The adapter executes bounded one-slot replacements	Eight of 20 full-evaluated candidates advance the frontier and seven enter winner lineages	Formation-stage development endpoint gains of $+0.72/+1.09$ points on mean-pass/strict-pass

C.5 Resource Accounting

Table 34 compares the resources used to reach AdaGEPA's $B = 560$ checkpoint with a paired GEPA default search through $B = 1120$ in the three SGD studies reported in the main text. Each arm includes its share of the common initial validation. The legal $B = 560$ prefixes ended at 544, 547, or 556 task-level metric calls, depending on which proposal steps completed before the budget cap. Thus, the table compares realized search prefixes, rather than equal-quality candidates or two methods run for the same number of task-level rollouts.

Table 34: Physical resources for AdaGEPA@$B = 560$ and the paired GEPA default@$B = 1120$ in the three selected SGD studies. Entries are per-run means for identifiable pairs, in AdaGEPA/default order. Tokens and configured USD include calls with recorded usage.

SGD study	Pairs	Successful requests	Tokens (M)	Configured USD
Original	5/5	550.8 / 1105.8	1.869 / 4.532	0.826 / 1.974
Extension	5/5	551.4 / 1103.6	1.728 / 4.195	0.758 / 1.828
10-run	9/10	548.0 / 1104.6	1.807 / 4.627	0.783 / 2.022

The first two studies use an ordering formed from five designated searches with the GEPA default: 5,522 successful requests, 22,490,786 recorded tokens, and $9.816482 in configured cost. The ten-run ordering adds five searches, bringing its formation total to 11,044 successful requests, 44,156,403 recorded tokens, and $19.168028 in configured cost. These formation resources are separate from the online comparisons in the table. In the ten-run study, one run has failed requests with unknown usage that cannot be assigned reliably to either side of the $B = 560$ prefix, so its resource pair is excluded from the exact prefix means; it remains in the quality analysis. The additional five formation searches also contain 51 failed requests with unknown usage. Neither set of events is counted as zero-cost usage. Configured USD uses the frozen technical price schedule, not a provider invoice.

The SGD unseen-service evaluation completed all five jobs and 60 logical checkpoint references across five seeds, three arms, and four checkpoints. After identical checkpoint prompts were deduplicated, 41 unique candidates were evaluated on the 104-dialogue panel. The evaluation used 4,264 Task-LM calls and no Reflection-LM calls. All calls completed without failure or truncation; its configured cost was $7.1364.

For the supplementary MultiWOZ $B = 560$ search-external evaluation, all ten candidates have complete 104-dialogue records. Two reused historical evaluations, while the other eight required 832 new successful Task-LM calls. There were no new failed callback events, truncated outputs, or Reflection-LM calls.

In the MultiWOZ development attribution comparison, service-aware allocation used a mean of 1094.0 metric calls and approximately $2.050 in configured cost per run; Coverage-only used 1095.2 calls and approximately $2.039.

In the bounded one-slot AIME diagnostic, three successful Task-LM outputs reached the 16,000-token limit and were scored as returned.